Un marco Conceptual para la Formalización de *crosscutting* en Desarrollos Orientados a Aspectos

J.M. Conejero, K.G. van den Berg y J. Hernández

Resumen— **El concepto de *crosscutting* suele ser descrito en términos de *scattering* y *tangling*[1]. Sin embargo, la distinción entre estos términos no suele ser concisa, llevando a situaciones ambiguas. En muchas ocasiones se hace necesario tener definiciones más precisas, por ejemplo para determinadas áreas de investigación como la identificación formal de asuntos transversales o crosscutting concerns. Proponemos un marco de trabajo conceptual para la formalización de estos conceptos basado en lo que denominamos un patrón de crosscutting que representa la relación entre elementos de dos niveles diferentes, por ejemplo concerns y una representación de dichos concerns. Estos conceptos son definidos formalmente en términos del álgebra lineal y representados mediante matrices y operaciones con dichas matrices. De este modo, el concepto de crosscutting puede ser claramente diferenciado de los términos de scattering y tangling. La utilidad de las matrices de dependencias presentadas se ilustra mediante el análisis de crosscutting a lo largo de una serie de niveles de refinamiento, que puede ser formalizado en lo que hemos denominado la concatenación de patrones de crosscutting.**

***Palabras clave* – Desarrollo de Software Orientado a Aspectos, Scattering, Tangling, Crosscutting.**

I. Introducción

Uno de los principios clave del Desarrollo de Software Orientado a Aspectos es la Separación de Asuntos o Separación de Concerns (SOC). Un *concern* puede ser definido de una manera genérica como algo de interés para un proceso de ingeniería [7]. Relacionado con este principio, normalmente podemos encontrar el problema de los asuntos transversales o crosscutting concerns. El término crosscutting normalmente se describe en términos de los conceptos scattering y tangling, por ejemplo, un crosscutting concern puede ser definido como un concern que se encuentra disperso (scattering) y enmarado (tangling) con otros concerns debido a una mala modularización. Sin embargo, la diferencia entre estos términos no es concisa, llevando en muchos casos a situaciones ambiguas o de confusión, como se cita en [11]:

.. the term "crosscutting concerns" is often misused in two ways: To talk about a single concern, and to talk about concerns rather than representations of concerns. Consider "synchronization is a crosscutting concern": we don't know that synchronization is crosscutting unless we know what it crosscuts. And there may be representations of the concerns involved that are not crosscutting.

Cuando hablamos de orientación a aspectos, usamos conceptos basándonos en nuestra propia experiencia e intuición. Sin embargo, son necesarias definiciones precisas para facilitar la utilización de herramientas, tales como herramientas de minería o identificación de crosscutting concerns (veáse [10]). En este sentido, el objetivo de este trabajo es describir un marco de trabajo conceptual con definiciones precisas de scattering, tangling y crosscutting.

El resto del artículo se estructura como sigue: en la Sección II. se introduce el Patrón de Crosscutting junto con las definiciones de crosscutting, scattering y tangling. En la Sección III. se describe cómo representar y visualizar el crosscutting mediante matrices. En la Sección IV. discutimos la aplicación del Patron de Crosscutting a lo largo de varios niveles consecutivos, en lo que hemos denominado la concatenación de Patrones de Crosscutting. La Sección V. muestra la aplicación del marco conceptual en un caso de estudio. Finalmente, en las Secciones VI. y VII. presentamos trabajos relacionados y las conclusiones del artículo respectivamente.

II. 2. El Patrón de Crosscutting

En esta sección discutimos nuestra premisa para la definición del Patrón de Crosscutting y el Patrón propiamente dicho. Además, en esta sección se muestran las definiciones de crosscutting, scattering y tangling.

A. Generalización

Nuestra propuesta se basa en la siguiente sentencia: los términos de crosscutting, scattering y tangling sólo pueden ser definidos en término de “una cosa” con respecto a “otra cosa”: al menos dos niveles (o dominios o fases) están relacionados entre sí de alguna manera. Por ejemplo:

- El término *dominio* puede hacer referencia a, por un lado,

Trabajo financiado por AOSD-Europe Project IST-2-004349-NoE (ver [1]) y MEC TIN2005-09405-C02-02.

J. M. Conejero, Quercus Software Engineering Group, Universidad de Extremadura, 10071 Cáceres, Spain (e-mail: chemacm@unex.es).

K. G. van den Berg, Software Engineering Group, University of Twente, 7500 AE Enschede, Netherlands (e-mail: k.g.vandenberg@ewi.utwente.nl).

J. Hernández, Quercus Software Engineering Group, Universidad de Extremadura, 10071 Cáceres, Spain (e-mail: juanher@unex.es).

[1] Dada su amplia utilización en la comunidad de la orientación a aspectos, a lo largo de este artículo se utilizarán las palabras crosscutting, scattering y tangling en lugar de su traducción al castellano.

concerns y, por otro, representación de esos concerns [11], como se menciona en la cita de la introducción.

- El término *nivel* puede referirse a diferentes refinamientos en un Desarrollo Dirigido por Modelos (Model Driven Development [14]), por ejemplo entre un CIM, un PIM y un PSM.
- El término *fase* puede referirse a diferentes fases en un ciclo de desarrollo de software, tales como modelado de concerns, análisis de requisitos, diseño arquitectónico, diseño detallado o implementación.

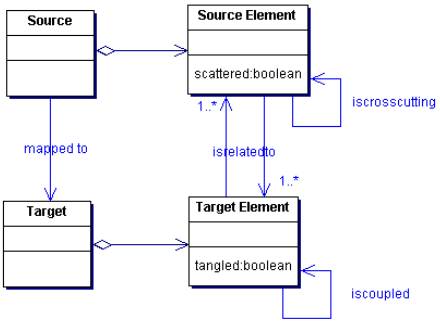

Fig. 1 Diagrama conceptual del Patrón de Crosscutting

Usamos los términos genéricos de source y target (como en [14]) para referirnos a estos dos niveles, dominios o fases consecutivas (ver Fig. 1) y el término de Patrón de Crosscutting para referirnos a esta situación en la que source y target están relacionados de alguna forma. Usamos el término de patrón de modo similar al que es utilizado en los patrones de diseño [9], en el sentido de ser una descripción general de una situación encontrada con cierta frecuencia [13], [15].

En el Patron de Crosscutting, los elementos del source están relacionados con elementos del target, es decir, existe una correspondencia entre dichos elementos. Esta relación puede ser establecida de modo manual o ser automatizada mediante reglas de transformación. A su vez, esta correspondencia puede ser de diferentes tipos, tales como utilización o dependencia abstracta (por ejemplo refinamiento y traza [17]). Los términos de crosscutting, tangling y scattering son definidos como casos específicos de la relación entre elementos del source y el target.

B. Conceptos Basados en el Patrón de Crosscutting

Como podemos ver en Fig. 1, existe una correspondencia entre elementos del source y el target, la cual tiene asociada una multiplicidad. En casos de relaciones de tipo 1:N, definimos scattering como sigue: *Scattering ocurre cuando en una correspondencia entre source y target, un elemento del source está relacionado con varios elementos del target.* Dada la correspondencia entre dos dominios, Source y Target, y definida del siguiente modo: para un elemento $s \in Source$, $f(s) = \{t \in Target / el\ elemento\ t\ está\ relacionado\ con\ s\}$, podemos definir también scattering de la siguiente forma: Un elemento $s \in Source$ está disperso sii $card(f(s)) > 1$, donde *card* es la cardinalidad de dicha relación.

De manera análoga, podemos centrarnos en la relación entre elementos del target y elementos del source. Esta relación es la relación inversa de la correspondencia anterior. En caso de relaciones de tipo 1:N entre target y source, definimos tangling del siguiente modo: *Tangling ocurre cuando, en una correspondencia entre source y target, un elemento del target está relacionado con varios elementos del source.* En otras palabras, un elemento $t \in Target$ está enmarañado sii $card(f^{1}(t)) > 1$, donde f^{1} es la inversa de f.

Existe una combinación especial de scattering y tangling que denominamos crosscutting. De este modo, definimos crosscutting como: *Crosscutting ocurre cuando, en una correspondencia entre source y target, un elemento del source está disperso (scattered) sobre varios elementos del target y, en al menos uno de esos elementos del target, otro elemento del source está mezclado (tangled).* En otras palabras, crosscutting puede ser definido como sigue: sean s1, s2 dos elementos distintos del Source. Entonces, *s1 corta (crosscuts) a s2 sii* $card(f(s1)) > 1$ *and* $\exists\, t \in f(s1)$: $card(f^{1}(t)) > 1$ *y* $s2 \in f^{1}(t)$. Como puede verse, nuestra definición no requiere que el segundo elemento del source esté disperso para tener crosscutting. En este sentido, nuestra definición no es simétrica, al contrario que la definición presentada en [13] (ver Sección VI.).

III. Representación Mediante Matrices

En términos del algebra lineal, la relación presentada entre source y target puede ser representada por medio de una matriz que hemos denominado matriz de dependencias. *Una matriz de dependencias (source x target) representa la correspondencia entre elementos del source y el target (relaciones de tipo entre-niveles).* En esta matriz, las filas representan los elementos del source, mientras que las columnas representan elementos del target. Un valor 1 en una celda de esta matriz denota la existencia de una relación entre los elementos source y target de la fila y columna correspondientes. Recíprocamente podríamos decir que el elemento del target de esa columna depende del elemento del source de esa fila. Por otro lado, esta matriz nos permite visualizar de una manera sencilla los elementos del source y del target que presentan scattering y tangling respectivamente.

TABLA 1
MATRIZ DE DEPENDENCIAS Y DE CROSSCUTTING CON TANGLING, SCATTERING Y NO CROSSCUTTING

matriz de dependencias

		Target				
		t[1]	t[2]	t[3]	t[4]	
source	s[1]	1	1	0	1	S
	s[2]	0	0	1	0	NS
	s[3]	0	0	1	0	NS
		NT	NT	T	NT	

matriz de crosscutting

		source		
		s[1]	s[2]	s[3]
source	s[1]	0	0	0
	s[2]	0	0	0
	s[3]	0	0	0

En Tabla 1, podemos observar un ejemplo de matriz de dependencia donde S en una fila indica que el elemento del source de esa fila está disperso (fila gris), NS indica no disperso, T en una columna indica que ese elemento del target se encuentra mezclado o enmarañado (columna gris) y NT

indica lo contrario.

A continuación definimos un nuevo concepto, denominado punto de corte (*crosscutpoint*), usado en el contexto de la matriz de dependencias para indicar una celda involucrada en scattering y tangling. Si existe uno o más *crosscutpoints* en una matriz de dependencias, entonces decimos que tenemos crosscutting.

Por último, utilizamos una nueva matriz, denominada matriz de crosscutting. *Una matriz de crosscutting (source x source) representa la relación de crosscutting entre elementos del source para una determinada correspondencia entre source y target (representada en una matriz de dependencias).* En dicha matriz, una celda con valor de 1 indica que el elemento del source de esa fila corta (crosscuts) al elemento source de la columna correspondiente. En la siguiente sección se explican los detalles necesarios para derivar la matriz de crosscutting a partir de la matriz de dependencias. En la Tabla 1, podemos observar la matriz de crosscutting para la matriz de dependencias mostrada en la misma tabla. En este ejemplo, existe un elemento del source s[1] disperso y un elemento del target t[3] enmarañado. Sin embargo, el elemento t[3] no se encuentra involucrado en el enmarañado de s[1]. Aplicando nuestra definición de crosscutting, obtenemos la matriz de crosscutting mostrada en esta tabla. Como podemos ver, en este caso, no existe crosscutting.

TABLA 2
MATRIZ DE DEPENDENCIAS Y DE CROSSCUTTING CON TANGLING, SCATTERING Y UN CROSSCUTPOINT

matriz de dependencias

		Target				
		t[1]	t[2]	t[3]	t[4]	
Source	s[1]	1	0	1	1	S
	s[2]	0	1	0	0	NS
	s[3]	0	0	1	0	NS
		NT	NT	T	NT	

matriz de crosscutting

		Source		
		s[1]	s[2]	s[3]
source	s[1]	0	0	1
	s[2]	0	0	0
	s[3]	0	0	0

En la Tabla 2 mostramos otro ejemplo de esta correspondencia entre source y target. En este caso existe un elemento del source s[1] disperso y un elemento del target t[3] enmarañado. Además, existe un *crosscutpoint* en la celda [1,3] (gris oscuro) de la matriz de dependencias. De nuevo, aplicando nuestra definición de crosscutting, obtenemos la matriz de crosscutting mostrada en la misma tabla. En este caso, el elemento del source s[1] corta al elemento s[3], ya que s[1] se encuentra disperso sobre [t[1],t[3],t[4]] y s[3] está enmarañado con s[1] en uno de esos elementos del target, en concreto t[3]. Por el contrario, s[3] no corta a s[1] puesto que no cumple las condiciones establecidas en la definición. Este caso ilustra cómo, según nuestra definición, la propiedad de crosscutting no es simétrica, s[1] corta a s[3], pero s[3] no corta a s[1] dado que s[3] no está disperso (scattering es una condición necesaria para tener crosscutting).

A. Construyendo las Matrices de Crosscutting

En esta sección describimos cómo derivar la matriz de crosscutting a partir de la matriz de dependencias. Para ilustrar la construcción de dichas matrices, utilizamos el ejemplo mostrado en la Tabla 2 de la sección anterior.

En primer lugar, basándonos en la matriz de dependencias, definimos dos matrices auxiliares: la *matriz de scattering* (source x target) y la *matriz de tangling* (target x source). Para el ejemplo mostrado en la Tabla 2, estas matrices se muestran en la Tabla 3.

TABLA 3
MATRIZ DE SCATTERING Y TANGLING PARA LA MATRIZ DE DEPENDENCIAS MOSTRADA EN TABLA 2

matriz de scattering

		target			
		t[1]	t[2]	t[3]	t[4]
Source	s[1]	1	0	1	1
	s[2]	0	0	0	0
	s[3]	0	0	0	0

matriz de tangling

		source		
		s[1]	s[2]	s[3]
target	t[1]	0	0	0
	t[2]	0	0	0
	t[3]	1	0	1
	t[4]	0	0	0

Estas matrices se construyen de la siguiente forma:

- En la matriz de scattering, una fila contiene las relaciones de dependencia entre elementos del source y el target si el elemento source de esa fila se encuentra disperso (relacionado con varios elementos del target); en otro caso, la fila contiene ceros (no existe scattering).
- En la matriz de tangling, una fila contiene las relaciones de dependencia entre elementos del target y el source si el elemento del target de esa fila se encuentra enmarañado (relacionado con varios elementos del source); en otro caso, la fila contiene sólo ceros (no existe tangling).

A continuación definimos la matriz de producto de crosscutting. *Una matriz de producto de crosscutting (source x source) representa la frecuencia de relaciones de crosscutting entre elementos del source para una determinada correspondencia entre source y target*. La matriz de producto de crosscutting ccpm puede obtenerse realizando el producto de las matrices de scattering sm y de tangling tm: *ccpm = sm x tm* donde *ccpm[i][k] = sm[i][j] x tm [j][k]*.

TABLA 4
MATRIX DE PRODUCTO DE CROSSCUTTING Y DE CROSSCUTTING PARA LA MATRIZ DE DEPENDENCIAS MOSTRADA EN TABLA 2

matriz de producto de crosscutting

		source		
		s[1]	s[2]	s[3]
Source	s[1]	1	0	1
	s[2]	0	0	0
	s[3]	0	0	0

matriz de crosscutting

		source		
		s[1]	S[2]	s[3]
source	s[1]	0	0	1
	s[2]	0	0	0
	s[3]	0	0	0

Como se ha comentado anteriormente, en esta matriz de producto de crosscutting, las celdas indican la frecuencia de crosscutting. Esta matriz puede ser utilizada para establecer métricas de crosscutting. En este ejemplo no existen celdas en la matriz con valores mayores que 1. A partir de esta matriz de producto de crosscutting, finalmente derivamos la matriz de crosscutting. Para ello, en primer lugar establecemos a 0 el valor de aquellas celdas de la diagonal que fueran distintas de 0. Evidentemente, estamos asumiendo que un elemento del source no puede cortarse a sí mismo.

En la matriz de crosscutting, las celdas indican la ocurrencia de crosscutting, independientemente de la

frecuencia. Además de la conversión realizada para aquellas celdas de la diagonal diferentes de 0, para derivar la matriz de crosscutting a partir de la matriz de producto de crosscutting, aplicamos otra simple conversión: *ccm[i][k] = if (ccpm[i][k] > 0) /\ (i ≠ k)* entonces 1, en otro caso, 0. Las matrices de producto de crosscutting y de crosscutting para el ejemplo evaluado a lo largo de esta sección se muestran en la Tabla 4. Por comodidad, estas formulas pueden ser calculadas de manera automática mediante simples herramientas matemáticas. Rellenando la matriz de dependencias, el resto de matrices son calculadas automáticamente.

IV. Concatenación de Patrones de Crosscutting

Normalmente, un proceso de desarrollo software consta de varios niveles o fases consecutivas. Por ejemplo, en MDA [14], podemos encontrar transformaciones entre Modelos Independientes de la Plataforma, Modelos Específicos de una Plataforma o Modelos de Implementación. Desde la perspectiva de las fases de un ciclo de vida software, podemos distinguir fases como Análisis del Dominio, Modelado de Concerns, Análisis de Requisitos, Diseño Arquitectónico, Diseño Detallado o Implementación.

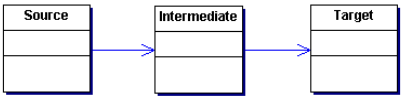

Fig. 2. Dos Patrones de Crosscutting concatenados

En esta sección consideramos la concatenación de dos Patrones de Crosscutting de modo que el dominio target del primer patrón sirve como dominio source para el segundo. (ver Fig. 2). Cada correspondencia que relaciona source y target en cada Patrón de Crosscutting puede describirse mediante una matriz de dependencias. A continuación indicamos cómo combinar dos matrices de dependencias consecutivas en una operación que hemos denominado concatenación. La operación de concatenación toma dos matrices de dependencias y genera como resultado una nueva matriz que representa la correspondencia existente entre los elementos del source de la primera matriz y los elementos del target de la segunda.

Tabla 5
Matrices de Dependencias de dos Patrones Concatenados

matriz de dependencias 1

	requisitos			
concern	r[1]	r[2]	r[3]	r[4]
c[1]	1	0	0	1
c[2]	0	1	0	0
c[3]	0	0	1	1

matriz de dependencias 2

	módulos				
requisitos	m[1]	m[2]	m[3]	m[4]	m[5]
r[1]	1	0	0	0	1
r[2]	0	1	0	0	0
r[3]	0	1	1	0	0
r[4]	0	0	0	1	1

Para esta concatenación, resulta esencial definir la transitividad de las relaciones de dependencia. Esta transitividad se define de la siguiente manera, dados los dominios source, intermedio y target y dados los elementos s ∈ source, t ∈ target:

$$(s\ R\ t) \text{ sii } \exists\ u \in \text{intermedio} / (s\ R\ u) \wedge (u\ R\ t)$$

La operación de concatenación puede generalizarse para varios niveles (n) mediante la composición de funciones [5]. Para ilustrar esta operación, mostramos un ejemplo de la concatenación de dos matrices de dependencias: una para concerns x requisitos y otra para requisitos x módulos de diseño. Estas dos matrices de dependencias se muestran en Tabla 5.

Dado que la primera matriz relaciona concerns con requisitos y la segunda requisitos con módulos de diseño, la matriz de dependencias resultante relaciona concerns con módulos de diseño (ver Tabla 6). Esta matriz se utiliza para derivar la matriz de crosscutting entre concerns (concern x concern) con respecto a los módulos de diseño (Tabla 6).

Tabla 6
Matriz de Dependencias Resultante y Matriz de Crosscutting Basadas en la Concatenación de las Matrices de la Tabla 5

matriz de dependencies resultante

	módulo				
concern	m[1]	m[2]	m[3]	m[4]	m[5]
c[1]	1	0	0	1	2
c[2]	0	1	0	0	0
c[3]	0	1	1	1	1

matriz de crosscutting

	concern		
concern	c[1]	c[2]	c[3]
c[1]	0	0	1
c[2]	0	0	0
c[3]	1	1	0

La concatenación de patrones de crosscutting puede ser utilizada para realizar análisis de trazabilidad a lo largo de varios niveles, por ejemplo desde concerns hasta elementos de implementación, pasando por requisitos, arquitectura o diseño (ver también [5]).

V. Case Study

En esta sección mostramos la aplicación de nuestra propuesta en un caso de estudio real. . El caso de estudio consiste en la implementación de un Sistema de Revisión de una Conferencia (CRS) [6] y ha sido utilizado en diversas conferencias como [8]. Por razones de espacio hemos simplificado este sistema eliminando funcionalidades que no interesaban para el ejemplo. El propósito principal del sistema consiste en asistir al comité de programa de una conferencia en las tareas de revisión de los artículos de la misma, así como permitir el registro en el sistema de los participantes de esa conferencia.

Existen cuatro tipos de usuario: *PcChair*, *PcMember*, *Author* y *Participant*. *PcChair* es el principal responsable del proceso de revisión llevado a cabo. Tiene acceso a todos los artículos y revisiones en el sistema. *PcMember* puede consultar información sobre los artículos y revisar los que le han sido asignados, pero no puede ver revisiones de otros *PcMember*. *Author* puede enviar un artículo al sistema y sólo puede consultar información sobre sus envíos. *Participant* debe registrarse en el sistema para poder asistir a la conferencia. El proceso de registro es diferente y está separado del proceso de autenticación en el sistema (*Login*).

Sin embargo, una vez que un usuario se ha registrado en el sistema, deberá autenticarse siempre que acceda al mismo. El proceso de autenticación chequea el rol del usuario para comprobar el tipo de acceso permitido. El diagrama de casos de uso del sistema se muestra en Fig. 3. En [6] podemos encontrar un análisis completo y más detallado de los requisitos de este sistema.

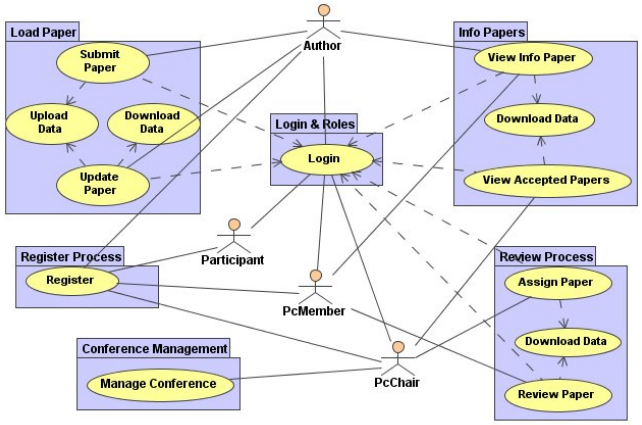

Fig. 3. Diagrama de Casos de Uso del Sistema de Revisión de una Conferencia

Teniendo en cuenta los requisitos del sistema, hemos identificado los siguientes concerns en el mismo: Papers Submission (PS), Papers Queries (PQ), Registration (Reg), Conference (C), Review (R), Information Retrieval/Supply (IRS), Login (L) y User Types (UT). Antes de poder aplicar nuestra propuesta en este ejemplo para identificar los crosscutting concerns, debemos elegir los elementos de los requisitos para analizar las correspondencias entre concerns y requisitos. Existen diferentes descomposiciones posibles, desde algunas de grano fino a otras de grano más grueso. En este caso hemos elegido los elementos principales del diagrama de casos de uso (cada paquete) mostrados en la Fig. 3 y el conjunto de actores que intervienen en el sistema como descomposición de los requisitos (grano grueso).

En Tabla 7a podemos observer la matriz de dependencias que representa las correspondencias entre concerns y requisitos, mientras que en Tabla 7b se muestra la matriz de crosscutting obtenida a partir de la primera. Como ya se ha comentado, existen otras descomposiciones posibles dando como resultado diferentes matrices de dependencias. Por ejemplo, podríamos haber usado una descomposición de requisitos de grano fino, tomando como elementos de la matriz cada caso de uso y cada actor por separado, con lo que el tamaño de la matriz se habría visto incrementado. En cualquier caso, la aplicación del marco de trabajo sería la misma y los resultados, a pesar de ser ligeramente diferentes, mostrarían que los concerns que cortan a otros serían también similares.

Como podemos observar en Tabla 7b, el concern de autenticación (*Login*) corta a todos los concerns donde el usuario debe realizar una operación de autenticación y el sistema debe chequear el rol del usuario. De manera análoga, el concern de acceso a la información (*Information Retrieval/Supply*) corta a los concerns que necesitan realizar acceso a la información almacenada para realizar las acciones oportunas.

TABLA 7.
(A) MATRIZ DE DEPENDENCIAS CONCERNS X REQS Y (B) MATRIZ DE CROSSCUTTING PARA EL CRS

(a) matriz de dependencias (concerns x requisitos)

		requisitos							
		Register Process	Info Papers	Load Papers	Review Process	Conf. Manag.	Login &Roles	Actors	
concerns	PS	0	0	1	0	0	0	0	NS
	PQ	0	1	0	0	0	0	0	NS
	Reg	1	0	0	0	0	0	0	NS
	C	0	0	0	0	1	0	0	NS
	R	0	0	0	1	0	0	0	NS
	IRS	1	1	1	1	1	0	0	S
	L	0	1	1	1	1	1	0	S
	UT	0	0	0	0	0	0	1	NS
		T	T	T	T	T	NT	NT	

(b) matriz de crosscutting (concerns x concerns)

		concerns							
		PS	PQ	Reg	C	R	IRS	L	UT
concerns	PS	0	0	0	0	0	0	0	0
	PQ	0	0	0	0	0	0	0	0
	Reg	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0
	IRS	1	1	1	1	1	0	1	0
	L	1	1	0	1	1	1	0	0
	UT	0	0	0	0	0	0	0	0

Una vez identificados los crosscutting concerns con respecto a los requisitos del sistema, podemos observar cómo se relacionan los concerns con el diseño del sistema.

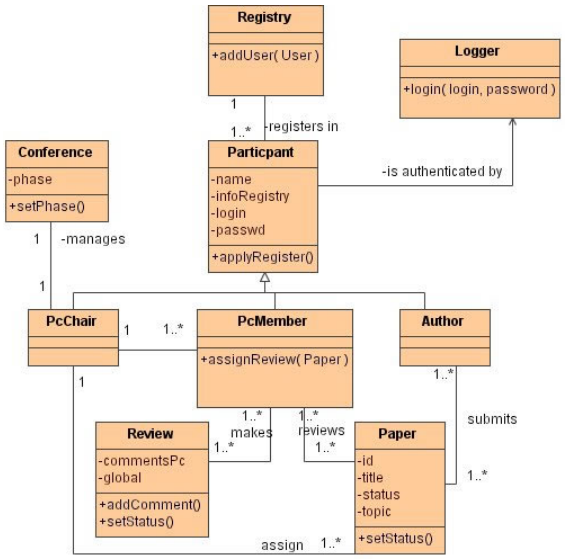

Fig. 4. Diagrama de Clases del CRS

En Fig. 4 mostramos un diagrama de clases UML que representa la estructura estática del diseño del sistema. A continuación, aplicamos nuestra propuesta tomando como entrada para la matriz de dependencias los requisitos representados en el diagrama de casos de uso (source) y las clases del diagrama de Fig. 4 (target). La matriz de dependencias construida puede verse en Tabla 8. En esta matriz mostramos la relación existente entre los requisitos del sistema y los elementos del diseño. Como podemos ver en Tabla 8, estas relaciones son directas excepto para los requisitos *Register Process* y *Login&Roles,* ya que se ha añadido cierta información en la clase *Participant* para realizar dicho registro y autenticación respectivamente (atributos *infoRegister, login* y *passwd* de dicha clase). Estos requisitos están mezclados en esta clase con la propia

funcionalidad de *Participant (User Type)*.

TABLA 8.
MATRIZ DE DEPENDENCIAS REQUISITOS X DISEÑO

		clases									
		Paper	Review	Conference	Pc Chair	Pc Member	Author	Participant	Logger	Registry	
requisitos	Register Process	0	0	0	0	0	0	1	0	1	S
	Info Papers	1	0	0	0	0	0	0	0	0	NS
	Load Papers	1	0	0	0	0	0	0	0	0	NS
	Review Process	0	1	0	0	0	0	0	0	0	NS
	Conf. Manag	0	0	1	0	0	0	0	0	0	NS
	Login& Roles	0	0	0	0	0	0	1	1	0	S
	Actors	0	0	0	1	1	1	1	0	0	S
		T	NT	NT	NT	NT	NT	T	NT	NT	

Ahora, aplicamos la operación de concatenación (definida en la Sección IV.) entre la matriz de dependencias para concerns x requisitos (Tabla 7a) y la matriz de dependencias para requisitos x diseño (Tabla 8). Esta operación nos permite obtener las relaciones entre concerns y los elementos del diseño. La matriz de dependencias resultante se muestra en Tabla 9a. Finalmente, aplicando nuestra definición de crosscutting para esta matriz de dependencias, podemos obtener la matriz de crosscutting mostrada en Tabla 9b. En esta última matriz podemos observar cómo existen nuevos crosscutting concerns -con respecto al diseño- que no aparecían en Tabla 7b. En este caso, el concern *Registration (Reg)* corta a los concerns *Information Retieval/Supply (IRS), Login (L)* and *User Types (UT)*. De manera análoga, el concern *User Types (UT)* corta a los concerns *Registration (Reg), Information Retieval/Supply (IRS)* y *Login (L)*. Como puede observarse en la matriz de dependencias resultante de la operación de concatenación (ver Tabla 9a), estos concerns están dispersos en varios elementos del diseño y, en al menos uno de estos elementos, existe funcionalidad de otros concerns mezclada.

TABLA 9
MATRIZ DE DEPENDENCIAS CONCATENADA CONCERNS X DESIGN (9A) Y MATRIZ DE CROSSCUTTING (9B)

matriz de dependencies concatenada

		clases									
		Paper	Review	Confe-rence	Pc Chair	Pc Member	Author	Partici-pant	Logger	Registry	
Concerns	PS	1	0	0	0	0	0	0	0	0	NS
	PQ	1	0	0	0	0	0	0	0	0	NS
	Reg	0	0	0	0	0	0	1	0	1	S
	C	0	0	1	0	0	0	0	0	0	NS
	R	0	1	0	0	0	0	0	0	0	NS
	IRS	2	1	1	0	0	0	1	0	1	S
	L	2	1	1	0	0	0	1	1	0	S
	UT	0	0	0	1	1	1	1	0	0	S
		T	T	T	NT	NT	NT	T	NT	T	

matriz de crosscutting

		concerns							
		PS	PQ	Reg	C	R	IRS	L	UT
concerns	PS	0	0	0	0	0	0	0	0
	PQ	0	0	0	0	0	0	0	0
	Reg	0	0	0	0	0	1	1	1
	C	0	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0
	IRS	1	1	1	1	1	0	1	1
	L	1	1	1	1	1	1	0	1
	UT	0	0	1	0	0	1	1	0

Evidentemente, las conclusiones obtenidas sobre el análisis de crosscutting dependen de la descomposición elegida en cada nivel y de las dependencias entre los elementos de dichos niveles. Existen varias alternativas que podrían ayudar a eliminar el crosscutting mediante el uso de una modularización o descomposición diferente (por ejemplo utilizando técnicas orientadas a aspectos como [3]). Con este caso de estudio hemos mostrado cómo analizar el crosscutting a lo largo de varias fases de un proceso de desarrollo software.

VI. TRABAJOS RELACIONADOS

Existen otras publicaciones donde se ha tratado la formalización de crosscutting. En [13], crosscutting es definido como sigue: *"For a pair of modules m_A and m_B we say that m_A crosscuts m_B with respect to X [the result domain] if and only if their projections onto X intersect, and neither of the projections is a subset of the other*." Siguiendo nuestra terminología, existe un dominio source formado por A y B y un target formado por X. La *intersección* en esta definición es similar a nuestro concepto de *crosscutpoint* (celda involucrada en scattering y tangling). Sin embargo, nuestra definición es menos restrictiva, ya que no requiere la relación de inclusión (*subset*): en nuestro caso, sólo requerimos que la cardinalidad de la proyección de m_A sobre X sea mayor que 1 (scattering), pero la cardinalidad de m_B sobre X puede ser mayor o igual a 1. Esta restricción también implica que en nuestra definición, el crosscutting no es tratado como una propiedad simétrica. Podemos hacer las mismas observaciones para la definición similar de crosscutting que puede encontrarse en [15].

Existen varios autores que han usado matrices (*design structure matrices DSM*) para analizar la modularidad en diseños software [2]. En [12], se muestra un método para dividir la matriz de estructura de diseño y mejorar la modularidad de los diseños orientados a objeto. Sin embargo, esta matriz de estructura de diseño representa dependencias de tipo intra-nivel y no dependencias inter-nivel como las representadas en nuestra matriz de dependencias. En [16] se describe una matriz de relaciones (concerns x requisitos) similar a la matriz de dependencias y es usada para identificar crosscutting concerns. Sin embargo, en este trabajo no existe una definición explícita de crosscutting.

Los trabajos descritos anteriormente carecen de una aplicación de sus definiciones de crosscutting a varios niveles consecutivos. En nuestro caso, hemos utilizado nuestra formalización para trazar los crosscutting concerns a través de varios niveles diferentes de un proceso de desarrollo software, como se ha mostrado en la operación de concatenación.

VII. CONCLUSIONES

En este artículo se ha propuesto un marco de trabajo conceptual para describir crosscutting. Hemos introducido el concepto de Patrón de Crosscutting que representa las correspondencias entre elementos de un conjunto source sobre elementos de un conjunto target. Utilizando source y target nos abstraemos de niveles o fases específicos del desarrollo de software. Hemos definido crosscutting, scattering y tangling

como casos específicos de esta relación entre source y target. Se han introducido las matrices de dependencias y crosscutting para visualizar estas definiciones, mostrando que es posible formalizar dichas definiciones.

El marco de trabajo presentado puede ser aplicado en áreas concretas de investigación tales como la identificación de crosscutting concerns en cualquier fase del desarrollo software [4] o la definición de métricas, por ejemplo para cuantificar la intensidad de crosscutting de un concern. Otro área de aplicación interesante es la operación de concatenación de Patrones de Crosscutting, que se utiliza para modelar relaciones de crosscutting a lo largo de varios niveles, por ejemplo modelado de concerns, requisitos, análisis arquitectónico, diseño detallado o implementación. De este modo, el marco de trabajo mejora el análisis de trazabilidad [5] a lo largo del desarrollo de un sistema.

VIII. Referencias

[1] AOSD-Europe (2005). AOSD Ontology 1.0 - Public Ontology of Aspect-Orientation. Retrieved May, 2005, from http://www.aosd-europe.net/documents/d9Ont.pdf.

[2] Baldwin, C.Y. & Clark, K.B. (2000). Design Rules vol I, The Power of Modularity. MIT Press.

[3] Baniassad, E. & Clarke, S. (2004). Theme: An Approach for Aspect-Oriented Analysis and Design. In 26th International Conference on Software Engineering, Edinburgh.

[4] Berg, K. van den, Conejero, J. M. & Hernández, J. (2006a). Identification of crosscutting in software design. In Aspect-Oriented Modeling Workshop at 5th AOSD, Bonn.

[5] Berg, K. van den, Conejero, J. M. & Hernández, J. (2006b). Analysis of Crosscutting across Software Development Phases based on Traceability. In Early Aspects Workshop at 28th ICSE, Shanghai.

[6] Cachero. C., Gómez, J., Párraga, A. & Pastor, O. (2001). Conference Review System: A Case of Study. In [8].

[7] Filman, R., et al. (2004). Aspect-Oriented Software Development. Addison-Wesley.

[8] First International Workshop on Web-Oriented Software Technology. (2001). http://www.dsic.upv.es/~west/iwwost01/ . Valencia.

[9] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns. Elements of reusable object-oriented software. Addison-Wesley.

[10] Hannemann, J. & Kiczales, G. (2001). Overcoming the Prevalent Decomposition in Legacy Code. In Workshop on Advanced Separations of Concerns, at ICSE May 2001, Toronto.

[11] Kiczales, G. Crosscutting. AOSD.NET Glossary 2005. At http://aosd.net/ wiki/index.php?title=Crosscutting.

[12] Lopes, C.V. and Bajracharya, S.K. (2005). An analysis of modularity in aspect oriented design. In 4th International Conference on Aspect-Oriented Software Development. 2005. Chicago.

[13] Masuhara, H. and Kiczales, G. (2003). Modeling Crosscutting in Aspect-Oriented Mechanisms. In 17th European Conference on Object Oriented Programming. 2003. Darmstadt.

[14] MDA (2003). MDA Guide Version 1.0.1, document number omg/2003-06-01.

[15] Mezini, M. and Ostermann, K. (2003). Modules for Crosscutting Models. In 8th International Conference on Reliable Software Technologies. Toulouse, France: LNCS 2655.

[16] Rashid, A., Moreira, A. and Araujo, J. (2003). Modularisation and Composition of Aspectual Requirements. In 2nd International Conference on Aspect Oriented Software Development. 2003. Boston.

[17] UML (2004). Unified Modeling Language 2.0 Superstructure Specification. At http://www.omg.org/cgi-bin/doc?ptc/2004-10-02

IX. Biografías

José M. Conejero nació en Mérida, España, el 10 de Abril de 1979. Obtuvo su título de Ingeniero en Informática por la Universidad de Extremadura en Septiembre de 2002.

Desde entonces ha trabajado como becario de Investigación en la Universidad de Extremadura con cargo a un proyecto del Ministerio de Ciencia y Tecnología. En este proyecto, trabaja en el modelado de aspectos y en la obtención de métodos para la identificación de aspectos en dominios concretos en fases de diseño y requisitos. Posteriormente ha trabajado en esta misma universidad como Profesor Ayudante impartiendo una asignatura relacionada con la Programación Orientada a Objetos. Sus principales campos de interés son la Programación y el Desarrollo de Software Orientado a Aspectos, así como el Desarrollo de Software Dirigido por Modelos.

Klaas van den Berg es profesor Titular en el Grupo de Ingeniería del Software de la Universidad de Twente (Holanda). Recibió su Licenciatura y Master en Ingeniería Eléctrica y posteriormente obtuvo un Doctorado en Ciencias de la Computación por la Universidad de Twente. Ha sido docente durante varios años en la Universidad de Zambia y la Universidad de Dar es Salaam. Actualmente, imparte cursos para de grado y post-grado de Ingeniería, Gestión y Arquitecturas Software en la Universidad de Twente. Sus áreas de interés incluyen las métricas para la calidad del software, modelado de software, evolución de software, trazabilidad, desarrollo de software orientado a aspectos y procesos de desarrollo dirigidos por modelos. Como investigador consolidado, ha dirigido o participado en varios proyectos de investigación relacionados con estas materias. Ha participado en varios talleres y conferencias internacionales ya sea como organizador o como miembro del comité de programa.

Juan Hernández es profesor Titular de la Universidad de Extremadura, donde actualmente dirige el Departamento de Informática, y es el investigador principal del grupo Quercus de Ingeniería del Software de esa Universidad. Obtuvo su grado de Doctor en Informática en la Universidad Politécnica de Madrid en 1995. Previamente había obtenido su Licenciatura en Matemáticas en la Universidad de Extremadura. Ha participado en diversas conferencias nacionales e internacionales tanto en el comité organizador como en el comité de programa, entre los que cabe destacar ECOOP'2002, Early-Aspects at OOPSLA'2005, CAiSE'2005, SEKE'2005, así como en múltiples ediciones de JISBD. Ha dirigido diversos proyectos del Plan Nacional de I+D+I, siendo el Desarrollo de Software Orientado a Aspectos su principal área de interés. En la actualidad es miembro del comité permanente de JISBD, así como miembro de ACM e IEEE.